

RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	000	FFFFFFFFFFF	FFFFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	000	FFFFFFFFFFF	FFFFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000000000	000	FFFFFFFFFFF	FFFFFFFFFFF
RRR	RRR	UUU	UUU	NNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNNNNN	NNN	000	000	FFF	FFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	000	FFFFFFFFF	FFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	000	FFFFFFFFF	FFFFFFFFF
RRRRRRRRRRRR		UUU	UUU	NNN	NNN	000	000	FFFFFFFFF	FFFFFFFFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNNNNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	000	FFF	FFF
RRR	RRR	UUU	UUU	NNN	NNN	000	000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU		NNN	NNN	000000000	000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU		NNN	NNN	000000000	000	FFF	FFF
RRR	RRR	UUUUUUUUUUUUUUUU		NNN	NNN	000000000	000	FFF	FFF

```

CCCCCCCCC LL HH HH
CCCCCCCCC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CC LL HH HH
CCCCCCCCC LLLLLLLLLL HH HH
CCCCCCCCC LLLLLLLLLL HH HH

```

[illegible]CLM  
VO4

.....

```
0001 0 %TITLE 'file processing interface and command line handler'
0002 0 MODULE clh ( IDENT = 'V04-000'
P 0003 0 %BLISS32 [ ADDRESSING_MODE (EXTERNAL = long_relative,
0004 0 NONEXTERNAL = long_relative)]
0005 0 ) =
0006 1 BEGIN
0007 1
0008 1 *****
0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS
33 0033 1
34 0034 1 ABSTRACT: File processing interface and command line handler.
35 0035 1
36 0036 1 ENVIRONMENT: Transportable
37 0037 1
38 0038 1 AUTHOR: R.W.Friday CREATION DATE: April, 1978
39 0039 1
```



```

41 0040 1 XSBTTL 'Revision History'
42 0041 1
43 0042 1 MODIFIED BY:
44 0043 1
45 0044 1 041 REM00041 Ray Marshall 23-February-1984
46 0045 1 Moved IN_TYPE and OUT_TYPE to GLBDAT.BLI and renamed them to
47 0046 1 IPFTOP and OPFTOP respectively. Also created there another
48 0047 1 global variable and one global literal to support new logic
49 0048 1 herein. All this was done because in VMS V4 there is a new
50 0049 1 feature (search lists) which necessitates our making calls
51 0050 1 to LIB$FIND FILE to find input files after the first one
52 0051 1 when multiple input files are specified. To do this, the
53 0052 1 logic to verify the input filespec was copied to a new
54 0053 1 routine within RUNOFF.BLI and changes were made herein to
55 0054 1 the sections that open the output and input files.
56 0055 1
57 0056 1 040 KFA00040 Ken Alden 20-Jul-1983
58 0057 1 Fixed wild card bug interaction with /auto.
59 0058 1
60 0059 1 039 KFA00039 Ken Alden 10-Jun-1983
61 0060 1 Improved error handling of wild-carding input file names.
62 0061 1
63 0062 1 038 KFA00038 Ken Alden 4-May-1983
64 0063 1 Added conditional for gca_skip_out in OUT_NO_CRLF branch.
65 0064 1
66 0065 1 037 KAD00037 Keith Dawson 14-Apr-1983
67 0066 1 Fixed bug: FLIP (.BFL) output file was not being opened.
68 0067 1
69 0068 1 036 KAD00036 Keith Dawson 22-Mar-1983
70 0069 1 Added LN01 support.
71 0070 1
72 0071 1 035 KFA00035 Ken Alden 07-Mar-1983
73 0072 1 Global edit of all modules. Updated module names, idents,
74 0073 1 copyright dates. Changed require files to BLISS library.
75 0074 1
76 0075 1 --
```

CLH  
V04-000

file processing interface and command line hand  
Module Level Declarations

K 14  
15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1 Page 3  
(3)

```

78 0076 1 %SBTTL 'Module Level Declarations'
79 0077 1
80 0078 1  TABLE OF CONTENTS:
81 0079 1
82 0080 1  FORWARD ROUTINE
83 0081 1      GET_OUT_DEFAULT : NOVALUE,
84 0082 1      BWAIT : NOVALUE,
85 0083 1      FBWAIT : NOVALUE;
86 0084 1
87 0085 1
88 0086 1  INCLUDE FILES:
89 0087 1
90 0088 1
91 0089 1  LIBRARY 'NXPORT:XPORT';           ! XPORT Library
92 0090 1  REQUIRE 'REQ:RNODEF';           ! RUNOFF variant definitions
93 0221 1
94 U 0222 1  %IF DSRPLUS %THEN
95 U 0223 1  LIBRARY 'REQ:DPLLIB';           ! DSRPLUS BLISS Library
96 0224 1  %ELSE
97 0225 1  LIBRARY 'REQ:DSRLIB';           ! DSR BLISS Library
98 0226 1  %FI
99 0227 1
100 0228 1
101 0229 1  MACROS:
102 0230 1
103 0231 1  MACRO
104 M 0232 1      XRPROMPT (TEXT) =
105 M 0233 1      ($XPO_PUT( IOB = TIOIOB
106 M 0234 1      ,STRING = ( %CHARCOUNT(TEXT)
107 0235 1      ,CH$PTR(UPLIT(TEXT))) ) ) %;
108 0236 1
109 0237 1
110 0238 1  OWN STORAGE:
111 0239 1
112 0240 1  OWN
113 0241 1      status;
114 0242 1
115 0243 1  OWN
116 0244 1      DEF_OUT_LNG,
117 0245 1      DEF_OUT_SPC : VECTOR [CH$ALLOCATION (50)];
118 0246 1
119 0247 1
120 0248 1  EXTERNAL REFERENCES:
121 0249 1
122 0250 1  EXTERNAL
123 0251 1      fra : FIXED_STRING,
124 0252 1      gca : GCA_DEFINITION,
125 0253 1      ts01 : FIXED_STRING,
126 0254 1      ira : FIXED_STRING,
127 0255 1      irac : IRAC_DEFINITION,
128 0256 1      ipftyp,           ! Cell in GLBDAT to hold the index into IPFTOP
129 0257 1      ipftop : VECTOR,
130 0258 1      opftop : VECTOR;
131 0259 1
132 0260 1  EXTERNAL
133 0261 1      IOBSTK : BLOCK,           ! IOB stack for doing .REQUIRE.
134 0262 1      RNEIOB : REF $XPO_IOB (), ! Always points to IOB for primary input.
```

CLH  
V04-000

file processing interface and command line hand  
Module Level Declarations

L 14  
15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1

Page 4  
(3)

```

: 135      0263 1      RNIOB : REF $XPO_IOB (),          !Primary input file
: 136      0264 1      RNOIOB : REF $XPO_IOB (),        !IOB for output file
: 137      0265 1      TSIOB : $XPO_IOB (),             !IOB for STREAM input from terminal.
: 138      0266 1      TTIOB : $XPO_IOB (),             !IOB for input from terminal
: 139      0267 1      TTOIOB : $XPO_IOB ();            !IOB for output to terminal
: 140      0268 1
: 141      0269 1      EXTERNAL LITERAL
: 142      0270 1      ipftct,      ! Literal defining the lengths of IPFTOP and OPFTOP.
: 143      0271 1
: 144      0272 1      ! Error messages
: 145      0273 1      rnfile,
: 146      U 0274 1      %IF dsrplus %THEN
: 147      U 0275 1      rnfoft,
: 148      0276 1      %FI
: 149      0277 1      rnfrtl;
: 150      0278 1
: 151      0279 1      EXTERNAL ROUTINE
: 152      0280 1      ERMS,
: 153      U 0281 1      %IF DSRPLUS %THEN
: 154      U 0282 1      ERM,
: 155      0283 1      %FI
: 156      0284 1      GRAB RESULTANT,
: 157      0285 1      PUTMSG,
: 158      0286 1      TSTTFE;
: 159      0287 1
```



CLH  
V04-000

file processing interface and command line hand M 14  
CLH -- OPCODE controlled main-line routine 15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742 Page 5  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1 (4)

```

161 0288 1 %sbttl 'CLH -- OPCODE controlled main-line routine'
162 0289 1 GLOBAL ROUTINE clh (opcode) =
163 0290 1
164 0291 1 ++
165 0292 1 FUNCTIONAL DESCRIPTION:
166 0293 1
167 0294 1 See ABSTRACT, above.
168 0295 1
169 0296 1 FORMAL PARAMETERS:
170 0297 1
171 0298 1 OPCODE specifies the operation to be performed.
172 0299 1
173 0300 1 IMPLICIT INPUTS: None
174 0301 1
175 0302 1 IMPLICIT OUTPUTS: None
176 0303 1
177 0304 1 ROUTINE VALUE:
178 0305 1 COMPLETION CODES:
179 0306 1
180 0307 1 See CLHCC.REQ for a description.
181 0308 1
182 0309 1 SIDE EFFECTS: None
183 0310 1
184 0311 1 --
185 0312 1
186 0313 2 BEGIN
187 0314 2
188 0315 2 LOCAL
189 0316 2 temp;
190 0317 2
191 0318 2 CASE .opcode FROM 1 TO clh_ops_count OF
192 0319 2 SET
193 0320 2
```

```
195 0321 2 %sbttl 'CLH -- open input file'
196 0322
197 0323 [CLH_OPEN_INPUT] : ! Open primary input file.
198 0324 BEGIN
199 0325
200 U 0326 %IF NOT %BLISS(BLISS32) AND DSRPLUS %THEN
201 U 0327
202 U 0328 INCR I FROM 0 TO (ipftct-1) DO ! fourteen different file types
203 U 0329 BEGIN ! loop until one found.
204 U 0330 IF .gca_pass_count GTR 1
205 U 0331 THEN
206 U 0332 BEGIN
207 U 0333 status = $XPO_OPEN (IOB = .rniob
208 U 0334 ,OPTIONS = INPUT
209 U 0335 ,DEFAULT = (4, .ipftop [ .ipftyp ])
210 U 0336 ,FAILURE = grab_resultant);
211 U 0337 IF .status
212 U 0338 THEN
213 U 0339 I = .ipftyp;
214 U 0340 END
215 U 0341 ELSE
216 U 0342 status = $XPO_OPEN (IOB = .rniob
217 U 0343 ,OPTIONS = INPUT
218 U 0344 ,DEFAULT = (4, .ipftop [.I ])
219 U 0345 ,FAILURE = grab_resultant);
220 U 0346
221 U 0347 IF .status THEN EXITLOOP ! If we open a file, exit the INCR loop
222 U 0348
223 U 0349 END; !end of INCR loop.
224 U 0350 %ELSE
225 P 0351 status = $XPO_OPEN (IOB = .rniob
226 P 0352 ,OPTIONS = INPUT
227 P 0353 ,DEFAULT = (4, CH$PTR (UPLIT ('.RNO'))))
228 P 0354 ,FAILURE = grab_resultant);
229 U 0355 %FI
230 U 0356 IF .status ! Succeeded in opening (any type)?
231 U 0357 THEN
232 U 0358 BEGIN ! Yes...
233 U 0359 BIND
234 U 0360 file_spec_stuff = rniob [iob$t_resultant] : $STR_DESCRIPTOR ();
235 U 0361
236 U 0362 ! Pick off the name and length of the file spec for ERROR.BLI:
237 U 0363
238 U 0364 irac_fspecp = .file_spec_stuff [str$a_pointer];
239 U 0365 irac_fspecc = .file_spec_stuff [str$h_length];
240 U 0366
241 U 0367 %IF NOT %BLISS(BLISS32) AND DSRPLUS %THEN
242 U 0368 ipftyp = .i;
243 U 0369 IF (.I GTR 0) !Succeeded in opening, but not '.RNO'.
244 U 0370 THEN
245 U 0371 ERM (rnfoft);
246 U 0372 %FI
247 U 0373 RETURN clh_normal
248 U 0374 END
249 U 0375 ELSE ! Open failed.
250 U 0376 RETURN clh_cant_open
251 U 0377 END;
```



```
253 0378 2 %sbtcl 'CLH -- open output file'
254 0379
255 0380
256 0381 [CLH_OPEN_OUT] :
257 0382   Open output file.
258 0383 BEGIN
259 0384   get_out_default (rniob [iob$$_resultant]);
260 0385   CASE .gca_op_dev FROM op_dev_first TO op_dev_last OF
261 0386
262 0387     SET
263 0388
264 0389     [op_dev_line_printer, op_dev_diablo] :
265 0390
266 0391       Normal cases: lineprinter output. Open STREAM output file using
267 0392       file type already determined (.MEM, .MEC, or whatever).
268 0393
269 0394       status = $XPO_OPEN
270 0395       (IOB = .rniob
271 0396        ,OPTIONS = OUTPUT
272 0397        ,ATTRIBUTES = STREAM
273 0398        ,RELATED = rniob [iob$$_resultant]
274 0399        ,DEFAULT = (.def_out_lng, CH$PTR (def_out_spc))
275 0400        ,FAILURE = grab_resultant
276 0401       );
277 0402   %IF DSRPLUS %THEN
278 0403       [op_dev_vt100] :
279 0404
280 0405       IF NOT .gca_s_output
281 0406       THEN
282 0407
283 0408         User said /DEC=VT100, and did not say /OUTPUT=name,
284 0409         so send output to TT: (SYS$OUTPUT).
285 0410
286 0411         status = $XPO_OPEN
287 0412         (IOB = .rniob
288 0413          ,OPTIONS = OUTPUT
289 0414          ,FILE_SPEC = $XPO_OUTPUT
290 0415          ,RELATED = rniob [iob$$_resultant]
291 0416          ,DEFAULT = (.def_out_lng, CH$PTR (def_out_spc))
292 0417          ,FAILURE = grab_resultant
293 0418         );
294 0419       ELSE
295 0420
296 0421         Open non-STREAM output file using file type already
297 0422         determined (.VT1).
298 0423
299 0424         status = $XPO_OPEN
300 0425         (IOB = .rniob
301 0426          ,OPTIONS = OUTPUT
302 0427          ,RELATED = rniob [iob$$_resultant]
303 0428          ,DEFAULT = (.def_out_lng, CH$PTR (def_out_spc))
304 0429          ,FAILURE = grab_resultant
305 0430         );
306 0431   %FI
307 0432   %IF LN01 %THEN
308 0433       [op_dev_ln01, op_dev_ln01e] :
309 0434
```

CLH  
V04-000

file processing interface and command line hand  
CLH -- open output file

C 15  
15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1

Page 8  
(6)

```

310      0435      !Open non-STREAM output file using file type already
311      0436      !determined (.LNI).
312      0437
313      0438      status = $XPO OPEN
314      0439      (IOB      = .rnoiob
315      0440      ,OPTIONS   = OUTPUT
316      0441      ,RELATED   = rniob [iob$t_resultant]
317      0442      ,DEFAULT   = (.def_out_lng, CH$PTR (def_out_spc))
318      0443      ,FAILURE   = grab_resultant
319      0444      );
320      0445      %FI
321      0446      %IF FLIP %THEN
322      0447      [op_dev_flip] :
323      0448      !Open non-STREAM output file using file type already
324      0449      !determined (.BFL).
325      0450
326      0451      status = $XPO OPEN
327      0452      (IOB      = .rnoiob
328      0453      ,OPTIONS   = OUTPUT
329      0454      ,RELATED   = rniob [iob$t_resultant]
330      0455      ,DEFAULT   = (.def_out_lng, CH$PTR (def_out_spc))
331      0456      ,FAILURE   = grab_resultant
332      0457      );
333      0458      %FI
334      0459      [INRANGE] : 0;
335      0460      [OUTRANGE] : 0;
336      0461
337      0462      TES;
338      0463
339      0464      IF NOT .status
340      0465      THEN
341      0466      RETURN clh_cant_open
342      0467
343      0468      ELSE
344      0469      RETURN clh_normal;
345      0470
346      0471      END;
```

```
0472 2 %sbttl 'CLH -- Read one record from current input file'
0473
0474 [CLH_READ_INPUT] :
0475 ! Read one record from current input file.
0476 BEGIN
0477 status = $XPO_GET (IOB = .rniob);
0478
0479 IF .status OR ! If no error in above get or
0480 (.status EQL xpo$_truncated) ! Truncated records are really not too bad.
0481 THEN
0482 ! A record was successfully read. Set up information
0483 ! needed by the remainder of the program.
0484 BEGIN
0485 irac_iseqn = .rniob [iob$_seq_numb]; ! Input record/sequence number.
0486 irac_ipagen = .rniob [iob$_page_numb]; ! Input page number.
0487 irac_seqn_flag = .rniob [iob$_sequenced]; ! Indicates meaning of IRAC_ISEQN.
0488
0489 ! Inform user if it was a truncated record that was read.
0490 IF .status EQL xpo$_truncated
0491 THEN
0492 erms (rnfrtl
0493 ..rniob [iob$_string]
0494 ,min (.rniob [iob$_h_string]
0495 ,50));
0496
0497 ! Set up the input as a FIXED_STRING.
0498 fs_start (ira) = .rniob [iob$_a_string];
0499 fs_next (ira) = .fs_start (ira);
0500 fs_maxsize (ira) = .rniob [iob$_h_string];
0501 fs_length (ira) = .fs_maxsize (ira);
0502
0503 !*****PATCH TO GET AROUND XPORT DEFICIENCIES
0504 ! Upon entering this block, the fixed string IRA is set up such that
0505 ! FS_NEXT(IRA) returns a CH$PTR to the first character to be processed.
0506 ! The contents of KHAR are undefined. The block is exited with the
0507 ! same conditions holding; the only effects are:
0508 ! 1. Updating the input page/line counters, and
0509 ! 2. Movement of FS_NEXT(IRA) over all LEADING formfeeds, nulls,
0510 ! and dels.
0511
0512 BEGIN
0513
0514 LITERAL
0515 ff = X0'014';
0516
0517 LOCAL
0518 ptr,
0519 x;
0520
0521 WHILE (.fs_length (ira) GTR 0) DO
0522 BEGIN
0523 ! First point to the character about to be considered.
0524 ptr = .fs_next (ira);
0525 ! Now, actually pick up the character. Note that
0526 ! FS_RCHAR is not used because it advances its pointer
0527 ! such that if this character is not to be discarded
```



```
405 0529 6      ! we can't back up.
406 0530 6      x = CH$RCHAR (.ptr);
407 0531 6
408 0532 6      SELECT .x OF
409 0533 6      SET
410 0534 6
411 0535 6      [FF] :
412 0536 7      BEGIN
413 0537 7      ! If the file is sequenced, LEADING formfeeds do not
414 0538 7      ! start new pages; the assumption here is that the
415 0539 7      ! file will be looked at using the same editor (presumably SOS)
416 0540 7      ! as the one that created it, and that editor behaves like
417 0541 7      ! SOS. The action for sequenced files is
418 0542 7      ! simply to ignore the formfeed.
419 0543 7      ! For unsequenced files, leading formfeeds do start new
420 0544 7      ! pages, especially if you look at that file using SOS.
421 0545 7      ! In such cases, XPORT does not pay attention to the
422 0546 7      ! formfeeds, and feeds them through without counting a
423 0547 7      ! new page. In this case, WE have to look for them and
424 0548 7      ! set up the page and sequence number items.
425 0549 7      ! (Note however that XPORT does count pagemarks).
426 0550 7
427 0551 7      IF NOT .rniob [iob$v_sequenced]
428 0552 7      THEN
429 0553 8      BEGIN
430 0554 8      irac_ipagen = .irac_ipagen + 1;
431 0555 8      irac_iseqn = 1;
432 0556 8      rniob [iob$g_seq_numb] = .irac_iseqn;
433 0557 8      rniob [iob$h_page_numb] = .irac_ipagen;
434 0558 7      END;
435 0559 7
436 0560 6      END;
437 0561 6
438 0562 6      [O, FF, DEL] :
439 0563 7      BEGIN
440 0564 7      ! Actually read the character that is being rejected.
441 0565 7      ! This results in FS_NEXT(IRA) pointing to the next
442 0566 7      ! character that is to be considered.
443 0567 7      fs_rchar (ira, x); ! (X is a dummy for this one line.)
444 0568 6      END;
445 0569 6
446 0570 6      [OTHERWISE] :
447 0571 6      EXITLOOP;
448 0572 6
449 0573 6      YES;
450 0574 6
451 0575 5      END;
452 0576 5
453 0577 4      END;
454 0578 4      !*****END OF PATCH
455 0579 4
456 0580 4      RETURN clh_normal;
457 0581 4      END
458 0582 3      ELSE
459 0583 4      BEGIN
460 0584 4
461 0585 4      IF .status EQL xpo$_end_file
```

CLH  
V04-000

file processing interface and command line hand F 15  
CLH -- Read one record from current input file 15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1

Page 11  
(7)

```

462 0586 4
463 0587 5
464 0588 5
465 0589 5
466 0590 5
467 0591 5
468 0592 6
469 0593 6
470 0594 6
471 0595 6
472 0596 6
473 0597 6
474 0598 6
475 0599 6
476 0600 6
477 0601 6
478 0602 6
479 0603 6
480 0604 6
481 0605 6
482 0606 6
483 0607 6
484 0608 5
485 0609 5
486 0610 5
487 0611 5
488 0612 5
489 0613 5
490 0614 4
491 0615 4
492 0616 5
493 0617 5
494 0618 5
495 0619 4
496 0620 4
497 0621 3
498 0622 3
499 0623 2
500 0624 2

      THEN
      BEGIN
      IF .gca_req_depth NEQ 0
      THEN
      ! It's a ".REQUIRE" file to be closed.
      BEGIN
      clh (clh_pop);
      Note that the following is a recursive call on this
      particular code sequence. When CLH encounters an end
      of file when it attempts to read a record from a
      require file, it must still, nevertheless, return a
      record, unless there is not more input. If a
      .REQUIRE command is the last record read from the
      file that referenced the file just closed, then the
      attempt to read a record from that file will also
      meet with an end of file being detected. In this
      case, you have to pop that file too, and try again.
      That happens until either a record is finally read,
      or all files have been popped and an end of file
      occurs when trying to read the main input file.
      RETURN clh (clh_read_input);
      END;

      ! End of file on primary input file.
      fs length (ira) = 0;
      RETURN clh_end_file;
      END
    ELSE
    ! Error reading input file.
    BEGIN
    fs length (ira) = 0;
    RETURN clh_cant_read;
    END;

  END;

END;
```

CLH  
V04

CLH  
V04-000

file processing interface and command line hand 15-Sep-1984 23:56:16  
CLH -- Write 1 record to O/P file with CRLF suf 14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1 Page 12  
(8)

```

0625 2 %sbttl 'CLH -- Write 1 record to O/P file with CRLF suffix'
0626
0627 [CLH_WRITE_OUT] :
0628 ! Write one record to the output file.
0629 BEGIN
0630
0631 XIF FLIP XTHEN
0632     LOCAL
0633         temp_record : $flip_rnotxt;
0634 XFI
0635
0636 ! Append carriage control information to the record
0637 fs_wchar (fra, %0'15'); !Carriage return
0638 fs_wchar (fra, %0'12'); !Line feed
0639
0640 XIF FLIP XTHEN
0641     IF (.gca_op_dev EQL op_dev_flip)
0642     THEN
0643     BEGIN
0644         temp_record [rnotxt_code] = flip$k_rnotxt;
0645         temp_record [rnotxt_length] = .fs_length (fra);
0646         CHSMOVE (.fs_length (fra)
0647             , .fs_start (fra)
0648             , CHSPTR (temp_record [rnotxt_text]));
0649         status = $XPO_PUT (IOB = .rnoiob,
0650             STRING = (.fs_length (fra)
0651                 , .fs_start (fra)
0652                 , CHSPTR (temp_record)));
0653     END
0654 ELSE
0655 XFI
0656
0657 status = $XPO_PUT (IOB = .rnoiob,
0658     STRING = (.fs_length (fra)
0659         , .fs_start (fra)));
0660
0661 ! Remove the appended characters from the end of the buffer
0662 fs_next (fra) = CH$PLUS (.fs_next (fra), -2); !Back up pointer
0663 fs_length (fra) = .fs_length (fra) - 2; !Back up counter
0664
0665 IF .status
0666 THEN
0667     RETURN clh_normal;
0668
0669 END;
0670
```



CLH  
V04-000

file processing interface and command line hand 15  
CLH -- Write 1 record to O/P file w/o CRLF suff 14-Sep-1984 13:05:41

M 15

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1 Page 13  
(9)

```

549      0671 2 %sbttl 'CLH -- Write 1 record to O/P file w/o CRLF suffix'
550      0672 2
551      0673 2 [CLH_OUT_NOCRLF] :
552      0674 2 T Write one record to the output file. Don't add carriage control information.
553      0675 2 BEGIN
554      0676 2 IF NOT .gca_skip_out
555      0677 2 THEN
556      0678 2 BEGIN
557      0679 4 %IF FLIP %THEN
558      0680 4 LOCAL
559      0681 4 TEMP_RECORD : $FLIP_RNOTXT;
560      0682 4
561      0683 4 IF (.gca_op_dev EQL op_dev_flip)
562      0684 4 THEN
563      0685 4 BEGIN
564      0686 4 TEMP_RECORD[RNOTXT_CODE] = FLIP$K_RNOTXT;
565      0687 4 TEMP_RECORD[RNOTXT_LENGTH] = .FS_LENGTH(FRA);
566      0688 4 CH$MOVE( .FS_LENGTH(FRA), .FS_START(FRA)
567      0689 4 ,CH$PTR(TEMP_RECORD[RNOTXT_TEXT]));
568      0690 4 STATUS = $XPO_PUT( IOB=.RNOIOB
569      0691 4 ,STRING=( 'FLIP$K_RNOTXT_BASISIZ+.FS_LENGTH(FRA)
570      0692 4 ,CH$PTR(TEMP_RECORD) ) );
571      0693 4 END
572      0694 4 ELSE
573      0695 4 %FI
574      0696 4
575      0697 4 STATUS = $XPO_PUT ( IOB = .RNOIOB
576      0698 4 ,STRING = ( .FS_LENGTH (FRA)
577      0699 4 ,.FS_START (FRA) ) );
578      0700 4 END;
579      0701 4
580      0702 4 IF .STATUS OR .gca_skip_out
581      0703 4 THEN
582      0704 4 RETURN CLH_NORMAL;
583      0705 4
584      0706 4 END;
585      0707 2
```

CLH  
V04-000

file processing interface and command line hand  
CLH -- file closing functions

I 15  
15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1

Page 14  
(10)

```

587      0708 2 %sbttl 'CLH -- file closing functions'
588      0709 [CLH_CLOSE_INPUT] :
589      0710 ! Close current input file.
590      0711 BEGIN
591      P 0712 STATUS = $XPO_CLOSE ( IOB = .RNOIOB
592      0713 ,FAILURE = grab_resultant);
593      0714 RETURN CLH_NORMAL;
594      0715 END;
595      0716
596      0717 [CLH_CLOSE_OUT] :
597      0718 ! Close output file
598      0719 BEGIN
599      P 0720 STATUS = $XPO_CLOSE ( IOB = .RNOIOB
600      0721 ,FAILURE = grab_resultant);
601      0722 RETURN CLH_NORMAL;
602      0723 END;
603      0724
604      0725 [CLH_CLOSE_DEL_OUT] :
605      0726 ! Close output file
606      0727 BEGIN
607      0728
608      0729 IF .RNOIOB[IOB$V_TERMINAL]
609      P 0730 THEN ! If it's a terminal,
610      0731 STATUS = $XPO_CLOSE ( IOB = .RNOIOB ! just close it.
611      0732 ,FAILURE = grab_resultant)
612      0733 ELSE
613      0734 BEGIN
614      P 0735 STATUS = $XPO_CLOSE ( IOB = .RNOIOB ! Otherwise, close
615      P 0736 ,OPTIONS = REMEMBER ! and delete it.
616      0737 ,FAILURE = grab_resultant);
617      P 0738 STATUS = $XPO_DELETE ( IOB = .RNOIOB
618      0739 ,FAILURE = grab_resultant);
619      0740 END;
620      0741
621      0742 RETURN CLH_NORMAL;
622      0743 END;
623      0744
```

CLH  
V04

CLH  
V04-000

file processing interface and command line hand J 15  
CLH -- Push IOB onto stack 15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742 Page 15  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1 (11)

```
.. 625 0745 2 %sbttl 'CLH -- Push IOB onto stack'
.. 626 0746 2
.. 627 0747 [CLH PUSH] :
.. 628 0748 BEGIN
.. 629 0749
.. 630 0750 If there are not too many files already open allocate
.. 631 0751 a new IOB on IOBSTK (pointed to by RNIIOB).
.. 632 0752
.. 633 0753 IF .GCA_REQ_DEPTH NEQ .GCA_MAX_REQUIRE
.. 634 0754 THEN
.. 635 0755 BEGIN
.. 636 0756 RNIIOB = IOBSTK + (IOB$K_LENGTH * %UPVAL) * .GCA_REQ_DEPTH;
.. 637 0757 GCA_REQ_DEPTH = .GCA_REQ_DEPTH + 1;
.. 638 0758 RETURN CLH_NORMAL;
.. 639 0759 END
.. 640 0760 ELSE ! No room on IOB stack for the IOB.
.. 641 0761 RETURN CLH_NO_SPACE;
.. 642 0762
.. 643 0763 END;
.. 644 0764 2
```



CLM  
V04-000

file processing interface and command line hand 15-Sep-1984 23:56:16  
CLM -- Pop IOB from stack and reaccess previous 14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1 Page 16  
(12)

```

646 0765 2 %sbttl 'CLM -- Pop IOB from stack and reaccess previous one'
647 0766
648 0767
649 0768 [CLM_POP] :
650 0769 ! Cause RUNOFF to stop reading from the current file
651 0770 ! and read from the previous file instead.
652 0771 BEGIN
653 0772
654 0773 IF .GCA_REQ_DEPTH EQL 0
655 0774 THEN
656 0775 BEGIN
657 0776 ! Internal logic error: should not try to
658 0777 ! pop the main file.
659 0778 0
660 0779 END
661 0780 ELSE
662 0781 BEGIN
663 0782 ! Forcefully terminate open .LIST, .NOTE and .IF commands that
664 0783 ! are still open when the file in which they occurred is about
665 0784 ! to be closed. I.E., do not allow .END commands to be in a
666 0785 ! different file than the the opening .LIST, .NOTE,... command.
667 0786 TSTTFE (.GCA_REQ_DEPTH);
668 0787
669 0788 IF .RNIIOB [IOB$V_OPEN] ! Really close the file if
670 0789 THEN ! there was a file opened
671 0790 ! ( see REQUIR.BLI for case when it's not )
672 0791 STATUS = $XPO_CLOSE ( IOB = .RNIIOB
673 0792 ! Now do pop the file stack. FAILURE = grab_resultant);
674 0793 !RNIIOB = .RNIIOB - (IOB$K_LENGTH * %UPVAL);
675 0794 GCA_REQ_DEPTH = .GCA_REQ_DEPTH - 1;
676 0795
677 0796 IF .GCA_REQ_DEPTH EQL 0
678 0797 THEN ! Popped all the way back to primary input file
679 0798 !RNIIOB = .RNEIOB; !Get real primary IOB.
680 0799
681 0800 ! The routine ERROR needs the following information in IRAC.
682 0801 BEGIN
683 0802 ! Pick of the name and length of the file spec
684 0803 BIND
685 0804 FILE_SPEC_STUFF = RNIIOB [IOB$T_RESULTANT] : $STR_DESCRIPTOR ();
686 0805 IRAC_FSPECP = .FILE_SPEC_STUFF [STR$A_POINTER];
687 0806 IRAC_FSPECC = .FILE_SPEC_STUFF [STR$H_LENGTH];
688 0807 END;
689 0808 IRAC_ISEQN = .RNIIOB [IOB$G_SEQ_NUMB];
690 0809 IRAC_IPAGEN = .RNIIOB [IOB$H_PAGE_NUMB];
691 0810 RETURN CLH_NORMAL;
692 0811 END;
693 0812
694 0813
695 0814 2
END;
```

```
697 0815 2 %sbttl 'CLH -- Open REQUIRE file spec.'
698 0816
699 0817
700 0818 [CLH_OPEN_REQ] :
701 0819 ! Open a file requested on a .REQUIRE command.
702 0820 BEGIN
703 0821 status = $XPO_IOB_INIT (IOB = .rniob);
704 0822 status = $XPO_OPEN (IOB = .rniob
705 0823 ,OPTIONS = INPUT, ! INPUT file
706 0824 ,DEFAULT = '.RNO', ! Default the extension only!
707 0825 ,FILE_SPEC = ( .fs_length(fs01) ! filename
708 0826 ,fs_start (fs01))
709 0827 ,FAILURE = grab_resultant);
710 0828
711 0829 IF .status
712 0830 THEN
713 0831 ! Reset input line/page counters.
714 0832 BEGIN
715 0833 irac_ipagen = 1;
716 0834 irac_iseqn = 1;
717 0835 BEGIN
718 0836 ! Pick off the name and length of the filespec.
719 0837 BIND
720 0838 file_spec_stuff = rniob [iob$t_resultant] : $STR_DESCRIPTOR ();
721 0839 irac_fspecp = .file_spec_stuff [str$a_pointer];
722 0840 irac_fspecc = .file_spec_stuff [str$h_length];
723 0841
724 0842 ! Output name of .REQUIRED file in .MEM file, if user
725 0843 said /DEBUG:FILES
726 0844
727 0845 IF .gca_debug_fil AND NOT .gca_skip_out
728 0846 THEN
729 0847
730 0848 ! Yes: User said /DEBUG:FILES and output is being
731 0849 ! generated because the current page is included
732 0850 ! in a /PAGES list.
733 0851
734 0852 BEGIN
735 0853 LOCAL
736 0854 %IF FLIP %THEN
737 0855 temp_record : $flip_rnotxt,
738 0856 %FI
739 0857 work_area : VECTOR [CH$ALLOCATION (100)],
740 0858 work_count,
741 0859 work_ptr;
742 0860
743 0861 work_ptr = CH$PTR (work_area);
744 0862 work_ptr = CH$MOVE (10, CH$PTR (UPLIT ('.REQUIRE ''')), .work_ptr);
745 0863 work_count = 10;
746 0864 work_ptr = CH$MOVE (.irac_fspecc, .irac_fspecp, .work_ptr);
747 0865 work_count = .work_count + .irac_fspecc;
748 0866 CH$WCHAR_A (XC'', work_ptr);
749 0867 CH$WCHAR_A (XO'15', work_ptr); !Carriage return
750 0868 CH$WCHAR_A (XO'12', work_ptr); !Line feed
751 0869 work_count = .work_count + 3;
752 0870 %IF FLIP %THEN
753 0871 IF (.gca_op_dev EQL op_dev_flip)
```

CLH  
V04-000

file processing interface and command line hand  
CLH -- Open REQUIRE file spec.

M 15  
15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1  
Page 18  
(13)

```

754      U 0872 6      THEN
755      U 0873 6      BEGIN
756      U 0874 6      temp_record [rnotxt_code] = flip$k_rnotxt;
757      U 0875 6      temp_record [rnotxt_length] = .work_count;
758      U 0876 6      CH$MOVE (.work_count, CH$PTR (work_area),
759      U 0877 6      CH$PTR (temp_record [rnotxt_text]));
760      U 0878 6      status = $XPO_PUT (IOB = .rnoiob,
761      U 0879 6      STRING = (flip$k_rnotxt_basesiz +
762      U 0880 6      .work_count,
763      U 0881 6      CH$PTR(temp_record)));
764      U 0882 6      END
765      U 0883 6      ELSE
766      U 0884 6      $XPO_PUT (IOB = .rnoiob,
767      P 0885 6      STRING = (.work_count, CH$PTR (work_area)));
768      U 0886 6      END;
769      U 0887 5      END;
770      U 0888 4      RETURN clh_normal;
771      U 0889 4
772      U 0890 4      END
773      U 0891 4      ELSE
774      U 0892 4      RETURN clh_cant_open;
775      U 0893 3
776      U 0894 3
777      U 0895 3
778      U 0896 2      END;
779      U 0897 2
```



file processing interface and command line hand N 15  
CLM -- Close REQUIRE file (NULL) and bad OPCode 15-Sep-1984 23:56:16 VAX-11 Bliss-32 V4.0-742 Page 19  
14-Sep-1984 13:05:41 DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1 (14)

```

781      0898      2 %sbttl 'CLM -- Close REQUIRE file (NULL) and bad OPCODE processing'
782      0899
783      0900      2 [CLM_CLOSE_REQ] :
784      0901      2 T Close a ".REQUIRE" file.
785      0902      2 BEGIN
786      0903      2 O
787      0904      2 END;
788      0905      2

```

**00000000000000000000000000000000**

```

790 0906 2 %SBTTL 'Open initialization file'
791 0907
792 0908 [CLH_OPEN_INIT] :
793 0909
794 0910 Open initialization file specified in fs01.
795 0911
796 0912 BEGIN
797 0913 status = $XPO_IOB_INIT (IOB = .rniob);
798 0914 status = $XPO_OPEN
799 0915 (IOB = .rniob,
800 0916 OPTIONS = input,
801 0917 FILE_SPEC = (.fs_length (fs01),
802 0918 .fs_start (fs01)),
803 0919 FAILURE = grab_resultant);
804 0920
805 0921 IF .status
806 0922 THEN
807 0923 Succeeded in opening file.
808 0924
809 0925 BEGIN
810 0926
811 0927 Pick off name and length of filespec.
812 0928
813 0929 BIND
814 0930 file_spec_stuff = rniob [iob$_resultant] : $STR_DESCRIPTOR ();
815 0931
816 0932 irac_fspeccp = .file_spec_stuff [str$a_pointer];
817 0933 irac_fspecc = .file_spec_stuff [str$h_length];
818 0934
819 0935 Output name of initialization file in .MEM file, if user
820 0936 said /DEBUG:FILES
821 0937
822 0938 IF .gca_debug_fil AND NOT .gca_skip_out
823 0939 THEN
824 0940
825 0941 Yes, user said /DEBUG:FILES and output is being
826 0942 generated because the current page is included
827 0943 in a /PAGES list.
828 0944
829 0945 BEGIN
830 0946 LOCAL
831 0947
832 0948 %IF FLIP %THEN
833 0949 temp_record : $flip_rnotxt,
834 0950 %FI
835 0951 work_area : VECTOR [CH$ALLOCATION (100)],
836 0952 work_count,
837 0953 work_ptr;
838 0954
839 0955 work_ptr = CH$PTR (work_area);
840 0956
841 0957 Move descriptive text into work area.
842 0958 Identifies DSR$INIT if only 8 characters long.
843 0959 Otherwise, it must be for DSRPLUS$INIT.
844 0960
845 0961 IF .fs_length (fs01) EQL 8
846 0962 THEN
```

```

      BEGIN
      work_count = 15;
      work_ptr = CH$MOVE (.work_count,
                          CH$PTR (UPLIT ('DSR$INIT file ''')),
                          .work_ptr);
      END
    ELSE
      BEGIN
      work_count = 19;
      work_ptr = CH$MOVE (.work_count,
                          CH$PTR (UPLIT ('DSRPLUS$INIT file ''')),
                          .work_ptr);
      END;
      |
      | Add file name to work area.
      |
      work_ptr = CH$MOVE (.irac_fspecc, .irac_fspecp, .work_ptr);
      work_count = .work_count + .irac_fspecc;
      |
      | Add end-of-line characters to work area.
      |
      CH$WCHAR_A (%C'', work_ptr);
      CH$WCHAR_A (%O'15', work_ptr);      |Carriage return
      CH$WCHAR_A (%O'12', work_ptr);      |Line feed
      work_count = .work_count + 3;
      |
      | If FLIP, send initialization file info to user
      | in correct form for FLIP.
      |
      IF (.gca_op_dev EQL op_dev_flip)
      THEN
        BEGIN
          temp_record [rnotxt_code] = flip$k_rnotxt;
          temp_record [rnotxt_length] = .work_count;
          CH$MOVE (.work_count, CH$PTR (work_area),
                  CH$PTR (temp_record [rnotxt_text]));
          status = $XPO_PUT (IOB = .rnoiob,
                            STRING = (flip$k_rnotxt_basesiz +
                                      .work_count,
                                      CH$PTR (temp_record)));
        END
      ELSE
        |
        | Otherwise, send initialization file info to user
        | in DSRPLUS form.
        |
        $XPO_PUT (IOB = .rnoiob,
                  STRING = (.work_count, CH$PTR (work_area)));
      END;
      RETURN clh_normal;
    ELSE
      |
      | Couldn't find file to open.

```

```

      0963 6
      0964 6
      0965 6
      0966 6
      0967 6
      0968 6
      0969 6
      0970 6
      0971 6
      0972 6
      0973 6
      0974 6
      0975 6
      0976 6
      0977 6
      0978 6
      0979 6
      0980 6
      0981 6
      0982 6
      0983 6
      0984 6
      0985 6
      0986 6
      0987 6
      0988 6
      0989 6
      0990 6
      0991 6
      0992 6
      0993 6
      0994 6
      0995 6
      0996 6
      0997 6
      0998 6
      0999 6
      1000 6
      1001 6
      1002 6
      1003 6
      1004 6
      1005 6
      1006 6
      1007 6
      1008 6
      1009 6
      1010 6
      1011 6
      1012 6
      1013 6
      1014 6
      1015 6
      1016 6
      1017 6
      1018 6
      1019 6

```

XIF FLIP XTHEN

XFI

P

! End of CLM

.PSECT SCODES,NOWRT,2



[illegible]

0494

	6E		03	18	001B9	BLEQU	13\$		
			32	D0	001BB	MOVL	#50, (SP)		
		38	A0	DD	001BE	PUSHL	56(R0)	0493	
00000000G	EF	00000000G	8F	DD	001C1	PUSHL	#RNFRTL	0492	
	51		03	FB	001C7	CALLS	#3, ERMS		
	FC		69	D0	001CE	MOVL	RNIOB, R1	0498	
	AB	38	A1	D0	001D1	MOVL	56(R1), IRA		
	6B	FC	AB	D0	001D6	MOVL	IRA, IRA+4	0499	
	04	34	A1	3C	001DA	MOVZWL	52(R1), IRA+8	0500	
	08	04	AB	D0	001DF	MOVL	IRA+8, IRA+12	0501	
		08	AB	D5	001E4	TSTL	IRA+12	0521	
	54		49	15	001E7	BLEQ	19\$		
	50		6B	D0	001E9	MOVL	IRA+4, PTR	0525	
	52		64	9A	001EC	MOVZBL	(PTR), X	0530	
	53		50	D0	001EF	MOVL	X, R2	0532	
	0C		01	D0	001F2	MOVL	#1, R3		
			52	D1	001F5	CMPL	R2, #12	0535	
			18	12	001F8	BNEQ	16\$		
11	30	A1	53	D4	001FA	CLRL	R3		
			04	E0	001FC	BBS	#4, 48(R1), 16\$	0551	
	F4	AB	F8	A8	D6	INCL	IRAC+12	0554	
	4C	A1	01	D0	00204	MOVL	#1, IRAC+8	0555	
	4A	A1	F4	A8	D0	MOVL	IRAC+8, 76(R1)	0556	
			F8	A8	B0	MOVW	IRAC+12, 74(R1)	0557	
			52	D5	00212	TSTL	R2	0562	
			0E	13	00214	BEQL	17\$		
	0C		52	D1	00216	CMPL	R2, #12		
0000007F	8F		09	13	00219	BEQL	17\$		
			52	D1	0021B	CMPL	R2, #127		
			0B	12	00222	BNEQ	18\$		
			53	D4	00224	CLRL	R3		
	50	00	BB	9A	00226	MOVZBL	@IRA+4, X	0567	
		08	6B	D6	0022A	INCL	IRA+4		
	B2		53	E9	0022C	DECL	IRA+12		
			0369	31	0022F	BLBC	R3, 15\$	0570	
00209000	8F		51	D1	00232	BRW	52\$	0583	
			18	12	00235	CMPL	R1, #2134016	0585	
			6A	D5	0023C	BNEQ	22\$		
			0F	13	00240	TSTL	GCA+188	0589	
			08	DD	00242	BEQL	21\$		
FDB7	CF		01	FB	00244	PUSHL	#8	0593	
			05	DD	00249	CALLS	#1, CLH		
FDB0	CF		01	FB	0024B	PUSHL	#5	0607	
			04	04	00250	CALLS	#1, CLH		
	50		02	D0	00251	RET			
			03	11	00254	MOVL	#2, R0	0616	
	50		03	D0	00256	BRB	23\$		
		08	AB	D4	00259	MOVL	#3, R0	0618	
				04	0025C	CLRL	IRA+12	0611	
00000000G	FF	00000000G	0D	90	0025D	RET		0583	
		00000000G	EF	D6	00264	MOVB	#13, @FRA+4	0637	
			EF	D6	0026A	INCL	FRA+4		
00000000G	FF	00000000G	0A	90	00270	INCL	FRA+12		
		00000000G	EF	D6	00277	MOVB	#10, @FRA+4	0638	
		00000000G	EF	D6	0027D	INCL	FRA+4		
	50	00000000G	EF	D0	00283	INCL	FRA+12		
						MOVL	RNOIOB, R0	0659	

64	AE	00000000G	EF	B0	0028A	MOVW	FRA+12, \$IOB\$OUTPUT	
66	AE		0E	90	00292	MOVB	#14, \$IOB\$OUTPUT+2	
67	AE		01	90	00296	MOVB	#1, \$IOB\$OUTPUT+3	
68	AE	00000000G	EF	D0	0029A	MOVL	FRA, \$IOB\$OUTPUT+4	
44	A0	64	AE	9E	002A2	MOVAB	\$IOB\$OUTPUT, 68(R0)	
2C	A0		07	90	002A7	MOVB	#7, 44(R0)	
		00000000G	EF	9F	002AB	PUSHAB	XPOS\$FAILURE	
			7E	D4	002B1	CLRL	-(SP)	
			50	DD	002B3	PUSHL	R0	
00000000G	EF		03	FB	002B5	CALLS	#3, XPOS\$PUT	
	67		50	D0	002BC	MOVL	R0, STATUS	
00000000G	EF		02	C2	002BF	SUBL2	#2, FRA+4	0662
00000000G	EF		02	C2	002C6	SUBL2	#2, FRA+12	0663
	4A		67	E8	002CD	BLBS	STATUS, 28\$	0665
			02D3	31	002D0	BRW	54\$	
	3C	B4	AA	E8	002D3	BLBS	GCA+112, 27\$	0676
	50	00000000G	EF	D0	002D7	MOVL	RNOIOB, R0	0699
64	AE	00000000G	EF	B0	002DE	MOVW	FRA+12, \$IOB\$OUTPUT	
66	AE		0E	90	002E6	MOVB	#14, \$IOB\$OUTPUT+2	
67	AE		01	90	002EA	MOVB	#1, \$IOB\$OUTPUT+3	
68	AE	00000000G	EF	D0	002EE	MOVL	FRA, \$IOB\$OUTPUT+4	
44	A0	64	AE	9E	002F6	MOVAB	\$IOB\$OUTPUT, 68(R0)	
2C	A0		07	90	002FB	MOVB	#7, 44(R0)	
		00000000G	EF	9F	002FF	PUSHAB	XPOS\$FAILURE	
			7E	D4	00305	CLRL	-(SP)	
			50	DD	00307	PUSHL	R0	
00000000G	EF		03	FB	00309	CALLS	#3, XPOS\$PUT	
	67		50	D0	00310	MOVL	R0, STATUS	
	04		67	E8	00313	BLBS	STATUS, 28\$	0702
	B6	B4	AA	E9	00316	BLBC	GCA+112, 25\$	
			7A	11	0031A	BRB	36\$	0704
	50		69	D0	0031C	MOVL	RNOIOB, R0	0713
			07	11	0031F	BRB	31\$	
	50	00000000G	EF	D0	00321	MOVL	RNOIOB, R0	0721
2C	A0		02	90	00328	MOVB	#2, 44(R0)	
		00000000G	EF	9F	0032C	PUSHAB	GRAB_RESULTANT	
			7E	D4	00332	CLRL	-(SP)	
			50	DD	00334	PUSHL	R0	
00000000G	EF		03	FB	00336	CALLS	#3, XPOS\$CLOSE	
			54	11	0033D	BRB	35\$	
	52	00000000G	EF	D0	0033F	MOVL	RNOIOB, R2	0729
10	32	A2	04	E1	00346	BBC	#4, 50(R2), 34\$	
2C	A2		02	90	0034B	MOVB	#2, 44(R2)	0732
		00000000G	EF	9F	0034F	PUSHAB	GRAB_RESULTANT	
			7E	D4	00355	CLRL	-(SP)	
			52	DD	00357	PUSHL	R2	
			DB	11	00359	BRB	32\$	
	2E	A2	10	88	0035B	BISB2	#16, 46(R2)	0737
2C	A2		02	90	0035F	MOVB	#2, 44(R2)	
		00000000G	EF	9F	00363	PUSHAB	GRAB_RESULTANT	
			7E	D4	00369	CLRL	-(SP)	
			52	DD	0036B	PUSHL	R2	
00000000G	EF		03	FB	0036D	CALLS	#3, XPOS\$CLOSE	
	67		50	D0	00374	MOVL	R0, STATUS	
	50	00000000G	EF	D0	00377	MOVL	RNOIOB, R0	0739
2C	A0		03	90	0037E	MOVB	#3, 44(R0)	
		00000000G	EF	9F	00382	PUSHAB	GRAB_RESULTANT	



			7E	D4	00388	CLRL	-(SP)	
			50	DD	0038A	PUSHL	R0	
00000000G	EF		03	FB	0038C	CALLS	#3, XPOSDELETE	
	67		50	DO	00393	35%:	MOVL	R0, STATUS
			79	11	00396	36%:	BRB	43\$
FC	AA		6A	D1	00398	37%:	CMPL	GCA+188, GCA+184
			14	13	0039C		BEQL	38\$
50	6A	000000F4	8F	C5	0039E		MULL3	#244, GCA+188, R0
	69	00000000G	40	9E	003A6		MOVAB	IOBS[K(R0)], RNIIOB
			6A	D6	003AE		INCL	GCA+188
			5F	11	003B0		BRB	43\$
	50		05	DO	003B2	38%:	MOVL	#5, R0
				04	003B5		RET	
	50		6A	DO	003B6	39%:	MOVL	GCA+188, R0
			03	12	003B9		BNEQ	40\$
			01E8	31	003BB		BRW	54\$
			50	DD	003BE	40%:	PUSHL	R0
00000000G	EF		01	FB	003C0		CALLS	#1, TSTTFE
	50		69	DO	003C7		MOVL	RNIIOB, R0
	18	32	A0	E9	003CA		BLBC	50(R0), 41\$
2C	A0		02	90	003CE		MOVB	#2, 44(R0)
		00000000G	EF	9F	003D2		PUSHAB	GRAB RESULTANT
			7E	D4	003D8		CLRL	-(SP)
			50	DD	003DA		PUSHL	R0
00000000G	EF		03	FB	003DC		CALLS	#3, XPOS\$CLOSE
	67		50	DO	003E3		MOVL	R0, STATUS
	69	000000F4	8F	C2	003E6	41%:	SUBL2	#244, RNIIOB
			6A	D7	003ED		DECL	GCA+188
			07	12	003EF		BNEQ	42\$
	69	00000000G	EF	DO	003F1		MOVL	RNEIOB, RNIIOB
	50		69	DO	003F8	42%:	MOVL	RNIIOB, R0
	51	1C	A0	9E	003FB		MOVAB	28(R0), R1
FC	A8	04	A1	DO	003FF		MOVL	4(R1), IRAC+16
	68		61	3C	00404		MOVZWL	(R1), IRAC+20
F4	A8	4C	A0	DO	00407		MOVL	76(R0), IRAC+8
F8	A8	4A	A0	32	0040C		CVTWL	74(R0), IRAC+12
			018A	31	00411	43%:	BRW	52\$
00F4	BF		69	DO	00414	44%:	MOVL	RNIIOB, R6
		00	00	2C	00417		MOVCS	#0, (SP), #0, #244, (R6)
			66		0041E			
	66	0301003D	8F	DO	0041F		MOVL	#50397245, (R6)
	A6	020E	8F	BO	00426		MOVW	#526, 30(R6)
1E	67	00208001	8F	DO	0042C		MOVW	#2129921, STATUS
	64	AE	00000000G	EF	BO	00433	MOVW	FS01+12, \$STR\$STRING
	66	AE		0E	90	0043B	MOVB	#14, \$STR\$STRING+2
	67	AE		01	90	0043F	MOVB	#1, \$STR\$STRING+3
	68	AE	00000000G	EF	DO	00443	MOVW	FS01, \$STR\$STRING+4
			7E	D4	0044B		CLRL	-(SP)
		68	AE	9F	0044D		PUSHAB	\$STR\$STRING
			7E	D4	00450		CLRL	-(SP)
00000000G	EF		03	FB	00452		CALLS	#3, XST\$FORMAT
	04	A6	50	DO	00459		MOVW	R0, 4(R6)
	08	A6	A7	9E	0045D		MOVAB	\$IOB\$DEFAULT, 8(R6)
	2E	A6	01	88	00462		BISB2	#1, 46(R6)
	2C	A6	01	90	00466		MOVB	#1, 44(R6)
		00000000G	EF	9F	0046A		PUSHAB	GRAB RESULTANT
			7E	D4	00470		CLRL	-(SP)

	00000000G	EF	56	DD	00472	PUSHL	R6		
		67	03	FB	00474	CALLS	#3, XPOSOPEN		
		03	50	D0	0047B	MOVL	R0, STATUS		
			67	E8	0047E	BLBS	STATUS, 45\$		0828
			011E	31	00481	BRW	53\$		
	F8	A8	01	D0	00484	45\$:	MOVL	#1, IRAC+12	0832
	F4	A8	01	D0	00488	MOVL	#1, IRAC+8		0833
50		69	1C	C1	0048C	ADDL3	#28, RNII0B, R0		0837
	FC	A8	04	A0	00490	MOVL	4(R0), IRAC+16		0839
		68	60	3C	00495	MOVZWL	(R0), IRAC+20		0840
03	B8	AA	03	E0	00498	BBS	#3, GCA+116, 47\$		0845
			00FE	31	0049D	46\$:	BRW	52\$	
		F9	B4	AA	E8	47\$:	BLBS	GCA+112, 46\$	
		53	08	AE	9E		MOVAB	WORK AREA, WORK_PTR	0861
63	00000000'	EF	0A	28	004A8	MOVCL3	#10, P.AAG, (WORK_PTR)		0862
		56	0A	D0	004B0	MOVL	#10, WORK_COUNT		0863
			00A5	31	004B3	BRW	51\$		0864
		56	69	D0	004B6	48\$:	MOVL	RNII0B, R6	0913
00F4	8F	00	6E	00	2C	MOVCL3	#0, (SP), #0, #244, (R6)		
			66		004C0				
	1E	66	0301003D	8F	D0	004C1	MOVL	#50397245, (R6)	
		A6	020E	8F	B0	004C8	MOVW	#526, 30(R6)	
		67	00208001	8F	D0	004CE	MOVL	#2129921, STATUS	
	64	AE	00000000G	EF	B0	004D5	MOVW	FS01+12, \$STR\$STRING	0919
	66	AE		0E	90	004DD	MOVB	#14, \$STR\$STRING+2	
	67	AE		01	90	004E1	MOVB	#1, \$STR\$STRING+3	
	68	AE	00000000G	EF	D0	004E5	MOVL	FS01, \$STR\$STRING+4	
				7E	D4	004ED	CLRL	-(SP)	
			68	AE	9F	004EF	PUSHAB	\$STR\$STRING	
				7E	D4	004F2	CLRL	-(SP)	
	00000000G	EF	03	FB	004F4	CALLS	#3, XST\$FORMAT		
	04	A6	50	D0	004FB	MOVL	R0, 4(R6)		
	2E	A6	01	88	004FF	BISB2	#1, 46(R6)		
	2C	A6	01	90	00503	MOVB	#1, 44(R6)		
			00000000G	EF	9F	00507	PUSHAB	GRAB RESULTANT	
				7E	D4	0050D	CLRL	-(SP)	
				56	DD	0050F	PUSHL	R6	
	00000000G	EF	03	FB	00511	CALLS	#3, XPOSOPEN		
		67	50	D0	00518	MOVL	R0, STATUS		
		03	67	E8	0051B	BLBS	STATUS, 49\$		0920
			0081	31	0051E	BRW	53\$		
50		69	1C	C1	00521	49\$:	ADDL3	#28, RNII0B, R0	0930
	FC	A8	04	A0	00525	MOVL	4(R0), IRAC+16		0932
		68	60	3C	0052A	MOVZWL	(R0), IRAC+20		0933
6C	B8	AA	03	E1	0052D	BBC	#3, GCA+116, 52\$		0938
		68	B4	AA	E8	00532	BLBS	GCA+112, 52\$	
		53	08	AE	9E	00536	MOVAB	WORK AREA, WORK_PTR	0955
		08	00000000G	EF	D1	0053A	CMPL	FS01+12, #8	0961
				0D	12	00541	BNEQ	50\$	
		56	0F	D0	00543	MOVL	#15, WORK_COUNT		0964
63	00000000'	EF	56	28	00546	MOVCL3	WORK_COUNT, P.AAH, (WORK_PTR)		0967
			0B	11	0054E	BRB	51\$		0961
		56	13	D0	00550	50\$:	MOVL	#19, WORK_COUNT	0971
63	00000000'	EF	56	28	00553	MOVCL3	WORK_COUNT, P.AAH, (WORK_PTR)		0974
63	FC	B8	68	28	0055B	51\$:	MOVCL3	IRAC+20, IRAC+16, (WORK_PTR)	0979
		56	68	C0	00560	ADDL2	IRAC+20, WORK_COUNT		0980
		83	0D22	8F	B0	00563	MOVW	#3362, (WORK_PTR)+	0984

CLM  
V04-000

file processing interface and command line hand  
Open initialization file

K 16

15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLM.BLI;1

Page 29  
(15)

02	83	0A	90	00568	MOVB	#10, (WORK PTR)+	0986
03	56	03	C0	00568	ADDL2	#3, WORK COUNT	0987
04	50	EF	D0	0056E	MOVL	RN0108, R0	1011
44	6E	56	B0	00575	MOVW	WORK_COUNT, \$IOB\$OUTPUT	
2C	AE	0E	90	00578	MOVB	#14, \$IOB\$OUTPUT+2	
	AE	01	90	0057C	MOVB	#1, \$IOB\$OUTPUT+3	
	AE	AE	9E	00580	MOVAB	WORK AREA, \$IOB\$OUTPUT+4	
	A0	6E	9E	00585	MOVAB	\$IOB\$OUTPUT, 68(R0)	
	A0	07	90	00589	MOVB	#7, 44(R0)	
		EF	9F	0058D	PUSHAB	XPOS\$FAILURE	
		7E	D4	00593	CLRL	-(SP)	
00000000G	EF	50	DD	00595	PUSHL	R0	
	50	03	FB	00597	CALLS	#3, XPOS\$PUT	
		01	D0	0059E	MOVL	#1, R0	1020
			04	005A1	RET		
		02	D0	005A2	MOVL	#2, R0	
			04	005A5	RET		
		50	D4	005A6	CLRL	R0	1034
			04	005A8	RET		

; Routine Size: 1449 bytes, Routine Base: \$CODE\$ + 0000

; 919 1035 1

```
1036 1 %sbttl 'get_out_default -- compute output filename from input filename'
1037 1 ROUTINE get_out_default (file_descriptor) : NOVALUE =
1038 1
1039 1 ++
1040 1 FUNCTIONAL DESCRIPTION:
1041 1
1042 1     This routine uses the input file type to compute the
1043 1     output file type (i.e., sometimes called extension).
1044 1
1045 1 FORMAL PARAMETERS:
1046 1
1047 1     FILE_DESCRIPTOR is a string descriptor for the input file name.
1048 1
1049 1 IMPLICIT INPUTS:      None
1050 1
1051 1 IMPLICIT OUTPUTS:     None
1052 1
1053 1 ROUTINE VALUE:
1054 1 COMPLETION CODES:     None
1055 1
1056 1 SIDE EFFECTS:         None
1057 1
1058 1 --
1059 1
1060 1 BEGIN
1061 1
1062 1 BIND
1063 1     file_spec_stuff = .FILE_DESCRIPTOR : $STR_DESCRIPTOR ();
1064 1
1065 1 LOCAL
1066 1     file_specx,
1067 1     file_spec_lng,
1068 1     name_length,
1069 1     parse_spec_blk : $XPO_SPEC_BLOCK,
1070 1     ptr_u_ext,
1071 1     ptr_i,
1072 1     ptr_o,
1073 1     type_length,
1074 1     u_ext : VECTOR [CH$ALLOCATION (50)];
1075 1
1076 1     file_specx = .file_spec_stuff [STR$A POINTER];
1077 1     file_spec_lng = .file_spec_stuff [STR$H_LENGTH];
1078 1
1079 1 !The code conditionalized by the %BLISS32 lexical function is
1080 1 !to get around a bug introduced in version 2.0 of VAX/VMS.
1081 1 !Logical names now start with an underscore, and this mucks up
1082 1 !XPORT's file spec parsing.
1083 1 %IF %BLISS(%BLISS32) %THEN
1084 1     !Inform the builder of RUNOFF that this kludge is here.
1085 1     %MESSAGE ('BLISS32 Patch code included to strip "" from output filespec');
1086 1     !Ignore the first character of the file spec if it's
1087 1     !an underscore so XPORT; this lets XPORT work correctly.
1088 1     IF CH$RCHAR(.file_specx) EQL XC'_'
1089 1     THEN
1090 1         !It's an underscore. Do the kludge.
1091 1         BEGIN
1092 1             file_specx = CH$PLUS (.file_specx, 1);
```



```

978      1093      3      file_spec_lng = .file_spec_lng - 1;
979      1094      END;
980      1095      XFI
981      1096      !End of crock code.
982      1097
983      1098      !Parse the input file spec to get the specified type.
984      1099      $XPO_PARSE_SPEC ( SPEC_BLOCK = parse_spec_blk
985      1100      ,FILE_SPEC = ( .file_spec_lng
986      1101      ,.file_specx ) );
987      1102      !Start building the output file spec.
988      1103      !
989      1104      !First set CH$PTRs to the parsed input file name.
990      1105      BEGIN
991      1106      BIND
992      1107      temp = parse_spec_blk [XPOST_FILE_NAME] : $STR_DESCRIPTOR ();
993      1108
994      1109      name_length = .temp [STR$H_LENGTH];
995      1110      ptr_i = .temp [STR$A_POINTER];
996      1111      END;
997      1112
998      1113      !Set a CH$PTR to where the computed output file spec goes.
999      1114      ptr_o = CH$PTR (def_out_spc);
1000      1115      !Copy the input file name to the output file spec.
1001      1116
1002      1117      INCR i FROM 1 TO .name_length DO
1003      1118      CH$WCHAR_A (CH$RCHAR_A (ptr_i), ptr_o);
1004      1119
1005      1120      def_out_lng = .name_length;      !Set current length.
1006      1121      BEGIN
1007      1122      BIND
1008      1123      temp = parse_spec_blk [XPOST_FILE_TYPE] : $STR_DESCRIPTOR ();
1009      1124      type_length = .temp [STR$H_LENGTH];
1010      1125      ptr_i = .temp [STR$A_POINTER];
1011      1126      END;
1012      1127
1013      1128      !Translate the file type to upper case.
1014      1129      !Leave the result in U_EXT.
1015      1130      ptr_u_ext = CH$PTR (u_ext);      !CH$PTR to where file type goes when in upper case.
1016      1131      INCR i FROM 1 TO .type_length DO
1017      1132      BEGIN
1018      1133      LOCAL
1019      1134      temp;
1020      1135      temp = CH$RCHAR_A (ptr_i);
1021      1136
1022      1137      IF lower_letter (.temp)
1023      1138      THEN
1024      1139      !Convert lower case letter to upper case.
1025      1140      temp = upper_case (.temp);
1026      1141
1027      1142      !Put processed character into file type area.
1028      1143      CH$WCHAR_A (.temp, ptr_u_ext);
1029      1144      END;
1030      1145
1031      1146
1032      1147
1033      1148      ! Search through the various known input filetypes looking for a type that
1034      1149      ! matches what was given.
```

```
1035 1150 2 ptr_u_ext = .opftop [0]; !Assume it won't be found, and point to ".MEM".
1036 1151 2
1037 1152 2
1038 1153 2 IF .ipftyp EQL -1 THEN ! We haven't yet mapped against IPFTOP
1039 1154 2 BEGIN
1040 1155 2 INCR i FROM 0 TO (ipftct-1) DO
1041 1156 2
1042 1157 2 IF CH$EQL ( .type length
1043 1158 2 ,CH$PTR (u_ext)
1044 1159 2 ,4
1045 1160 2 ,.ipftop [i] )
1046 1161 2 THEN
1047 1162 2 !Found a match. Set a CH$PTR to the matching output file type.
1048 1163 2 BEGIN
1049 1164 2 ptr_u_ext = .opftop [i];
1050 1165 2 EXITLOOP
1051 1166 2 END
1052 1167 2
1053 1168 2 ELSE
1054 1169 2 ptr_u_ext = .opftop [.ipftyp];
1055 1170 2
1056 1171 2 U XIF DSRPLUS XTHEN
1057 1172 2 U IF .gca_op_dev EQL op_dev_vt100
1058 1173 2 U THEN
1059 1174 2 U !User said /DEC=VT100, so make .VT1 the default output type.
1060 1175 2 U ptr_u_ext = CH$PTR (UPLIT ('.VT1'));
1061 1176 2 U XFI
1062 1177 2
1063 1178 2 U XIF LN01 XTHEN
1064 1179 2 U IF (.gca_op_dev EQL op_dev_ln01
1065 1180 2 U OR .gca_op_dev EQL op_dev_ln01e)
1066 1181 2 U THEN
1067 1182 2 U !User said /DEVICE=LN01[e], so make .LNI the default output type.
1068 1183 2 U ptr_u_ext = CH$PTR (UPLIT ('.LNI'));
1069 1184 2 U XFI
1070 1185 2
1071 1186 2 U XIF FLIP XTHEN
1072 1187 2 U IF (.gca_op_dev EQL op_dev_flip)
1073 1188 2 U THEN
1074 1189 2 U !User said /DEC=FLIP; this overrides a /DEC=VT100 (if also given).
1075 1190 2 U ptr_u_ext = CH$PTR (UPLIT ('.BFL'));
1076 1191 2 U XFI
1077 1192 2
1078 1193 2 !Copy the file type to the output file spec area.
1079 1194 2 !Note that if there was no match, then PTR_U_EXT points to ".MEM"
1080 1195 2
1081 1196 2 INCR i FROM 1 TO 4 DO
1082 1197 2 CH$WCHAR_A (CH$RCHAR_A (ptr_u_ext), ptr_o);
1083 1198 2
1084 1199 2 def_out_lng = .def_out_lng + 4; !Update file spec length.
1085 1200 2 END; !End of get_out_default
```

.PSECT SPLITS,NOWRT,NOEXE,2

49 4E 4C 2E 0003C P.AAK: .ASCII \.LNI\

```
.EXTRN XPOS_PARSE_SPEC
.PSECT $CODE$,NOWRT,2

07FC 00000 GET_OUT_DEFAULT:
    .WORD Save_R2,R3,R4,R5,R6,R7,R8,R9,R10
    MOVAB GCA+208, R10
    MOVAB OPFTOP, R9
    MOVAB DEF_OUT_LNG, R8
    MOVAB -132(SPT, SP)
    MOVL FILE_DESCRIPTOR, R0
    MOVL 4(R0), FILE_SPECX
    MOVZWL (R0), FILE_SPEC_LNG
    CMPB (FILE_SPECX), #95
    BNEQ 1$
    INCL FILE_SPECX
    DECL FILE_SPEC_LNG
    MOVW FILE_SPEC_LNG, $STR$FILE_SPEC
    MOVAB #14, $STR$FILE_SPEC+2
    MOVAB #1, $STR$FILE_SPEC+3
    MOVL FILE_SPECX, $STR$FILE_SPEC+4
    PUSHAB XPOS_FAILURE
    CLRL -(SP)
    MNEGL #1, -(SP)
    PUSHAB PARSE_SPEC_BLK
    PUSHAB $STR$FILE_SPEC
    CALLS #5, XPOS_PARSE_SPEC
    MOVZWL TEMP, NAME_LENGTH
    MOVL TEMP+4, PTR_I
    MOVAB DEF_OUT_SPEC, PTR_O
    CLRL I
    BRB 3$
    MOVAB (PTR_I)+, (PTR_O)+
    AOBLEQ NAME_LENGTH, I, 2$
    MOVL NAME_LENGTH, DEF_OUT_LNG
    MOVZWL TEMP, TYPE_LENGTH
    MOVL TEMP+4, PTR_I
    MOVAB U_EXT, PTR_O_EXT
    CLRL I
    BRB 6$
    MOVZBL (PTR_I)+, TEMP
    CMPL TEMP, #97
    BLSS 5$
    CMPL TEMP, #122
    BGTR 5$
    SUBL2 #32, TEMP
    MOVAB TEMP, (PTR_U_EXT)+
    AOBLEQ TYPE_LENGTH, I, 4$
    MOVL OPFTOP, PTR_U_EXT
    MOVL IPFTYP, R0
    CMPL R0, #-1
    BNEQ 9$
    MNEGL #1, I
    BRB 8$
    MOVL IPFTOP[I], R0
    CMPC5 TYPE_LENGTH, U_EXT, #0, #4, (R0)

1037
1063
1076
1077
1088
1092
1093
1101
1109
1110
1114
1118
1120
1124
1125
1130
1131
1137
1139
1142
1145
1151
1153
1155
1160
1157
```

```

file processing interface and command line hand 0 1
get_out_default -- compute output filename from 15-Sep-1984 23:56:16
14-Sep-1984 13:05:41

```

VAX-11 Bliss-32 V4.0-742 Page 34  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI:1 (16)

.....

			60	000C7		BNEQ	8\$		
			06	12 000C8		MOVL	OPFTOP[I], PTR_U_EXT		1164
		55	6944	D0 000CA		BRB	10\$		1163
	E1		0E	11 0C0CE		AOBLEQ	#IPFTCT-1, I, 7\$		1157
		54	00000000G	8F F3 000D0	8\$:	BRB	10\$		1154
			04	11 000DB		MOVL	OPFTOP[R0], PTR_U_EXT		1169
04		55	6940	D0 000DA	9\$:	CMPZV	#4, #4, GCA+208, #4		1179
	6A	04	04	ED 000DE	10\$:	BEQL	11\$		
			07	13 000E3		CMPZV	#4, #4, GCA+208, #5		1180
05		6A	04	ED 000E5		BNEQ	12\$		
			07	12 000EA		MOVAB	P.AAK, PTR_U_EXT		1183
		55	00000000'	EF 9E 000EC	11\$:	MOVL	#1, I		1197
		50		01 D0 000F3	12\$:	MOVB	(PTR_U_EXT)+, (PTR_0)+		
		86		85 90 0U0F6	13\$:	AOBLEQ	#4, I, -13\$		
	F9	50		04 F3 000F9		ADDL2	#4, DEF_OUT_LNG		1199
		68		04 C0 000FD		RET			1200
				04 00100					

; Routine Size: 257 bytes, Routine Base: \$CODES + 05A9

: 1086 1201 1



```
1088 1202 1 %sbtll 'FBWAIT -- performs use synchronization for /PAUSE O/P w/FF'
1089 1203 1 GLOBAL ROUTINE fbwait : NOVALUE =
1090 1204 1
1091 1205 1 ++
1092 1206 1 FUNCTIONAL DESCRIPTION:
1093 1207 1
1094 1208 1 Issues some BELLS (^Gs) and a FORMFEED (^L) and waits for
1095 1209 1 the RUNOFF user to input a single character.
1096 1210 1
1097 1211 1 FORMAL PARAMETERS: None
1098 1212 1
1099 1213 1 IMPLICIT INPUTS: None
1100 1214 1
1101 1215 1 IMPLICIT OUTPUTS: None
1102 1216 1
1103 1217 1 ROUTINE VALUE:
1104 1218 1 COMPLETION CODES: None
1105 1219 1
1106 1220 1 SIDE EFFECTS: None
1107 1221 1
1108 1222 1 --
1109 1223 1
1110 1224 2 BEGIN
1111 P 1225 2 $XPO_GET (IOB = tsiob
1112 P 1226 2 ,PROMPT = (3, CH$PTR (UPLIT (
1113 P 1227 2 %STRING (BELL, BELL, %CHAR (%0'14'))))) )
1114 P 1228 2 ,CHARACTERS = 1
1115 1229 2 );
1116 1230 2
1117 1231 2 !Send a carriage return, so text starts at the left margin
1118 P 1232 2 $XPO_PUT (IOB = tsiob
1119 P 1233 2 ,STRING = (1, CH$PTR(UPLIT(%STRING(%CHAR(%0'15')))))
1120 1234 2 );
1121 1235 2
1122 1236 1 END; !End of FBWAIT
```

```
.PSECT $SPLITS,NOWRT,NOEXE,2
00 0C 07 07 00040 P.AAP: .ASCII <7><7><12><0>
00 00 00 0D 00044 P.AAT: .ASCII <13><0><0><0>
. EXTRN XST$FREE_TEMP
. PSECT $CODE$,NOWRT,2
02 54 00000000G EF 9E 00002 .ENTRY FBWAIT, Save R2,R3,R4 : 1203
03 53 00000000G EF 9E 00009 MOVAB XPOS$FAILURE, R4
04 5E 00000000G 08 C2 00010 MOVAB IOB$+36, R3
02 6E 00000000G 03 B0 00013 SUBL2 #8, SP
03 AE 00000000G 0E 90 00016 MOVW #3, $STR$STRING
04 AE 00000000G 01 90 0001A MOVW #14, $STR$STRING+2 : 1229
04 AE 00000000G EF 9E 0001E MOVW #1, $STR$STRING+3
04 AE 00000000G 7E D4 00026 MOVAB P.AAP, $STR$STRING+4
04 AE 9F 00028 CLRL -(SP)
PUSHAB $STR$STRING
```

00000000G	EF		7E	D4	0002B	CLRL	-(SP)	
	52		03	FB	0002D	CALLS	#3, XST\$FORMAT	
	50		50	D0	00034	MOVL	R0, R2	
			63	D0	00037	MOVL	IOB\$+36, R0	
			09	13	0003A	BEQL	1\$	
00000000G	EF		50	DD	0003C	PUSHL	R0	
	63		01	FB	0003E	CALLS	#1, XST\$FREE_TEMP	
10	A3		52	D0	00045	MOVL	R2, IOB\$+36	
12	A3		01	B0	00048	MOVW	#1, IOB\$+52	
08	A3		0E	90	0004C	MOVB	#14, IOB\$+54	
			06	90	00050	MOVB	#6, IOB\$+44	
			54	DD	00054	PUSHL	R4	
			7E	D4	00056	CLRL	-(SP)	
00000000G	EF	DC	A3	9F	00058	PUSHAB	IOB\$	
	6E		03	FB	0005B	CALLS	#3, XPOS\$GET	
02	AE		01	B0	00062	MOVW	#1, \$IOB\$OUTPUT	1234
03	AE		0E	90	00065	MOVB	#14, \$IOB\$OUTPUT+2	
04	AE	00000000'	01	90	00069	MOVB	#1, \$IOB\$OUTPUT+3	
20	A3		EF	9E	0006D	MOVAB	P.AAT, \$IOB\$OUTPUT+4	
08	A3		6E	9E	00075	MOVAB	\$IOB\$OUTPUT, IOB\$+68	
			07	90	00079	MOVB	#7, IOB\$+44	
			54	DD	0007D	PUSHL	R4	
			7E	D4	0007F	CLRL	-(SP)	
00000000G	EF	DC	A3	9F	00081	PUSHAB	IOB\$	
			03	FB	00084	CALLS	#3, XPOS\$PUT	
			04	0008B		RET		1236

; Routine Size: 140 bytes, Routine Base: \$CODE\$ + 06AA

; 1123 1237 1

```
1125 1238 1 %sbtll 'BWAIT -- performs user synchronization for /PAUSE 0/P w/o FF'
1126 1239 1 GLOBAL ROUTINE bwait : NOVALUE =
1127 1240 1
1128 1241 1 ++
1129 1242 1 FUNCTIONAL DESCRIPTION:
1130 1243 1
1131 1244 1 This routine is just like FBWAIT, except that no FORMFEED is issued.
1132 1245 1
1133 1246 1 FORMAL PARAMETERS: None
1134 1247 1
1135 1248 1 IMPLICIT INPUTS: None
1136 1249 1
1137 1250 1 IMPLICIT OUTPUTS: None
1138 1251 1
1139 1252 1 ROUTINE VALUE:
1140 1253 1 COMPLETION CODES: None
1141 1254 1
1142 1255 1 SIDE EFFECTS: None
1143 1256 1
1144 1257 1 --
1145 1258 1
1146 1259 2 BEGIN
1147 1260 2
1148 1261 2 EXTERNAL
1149 1262 2 tsiob: $XPO_IOB();
1150 1263 2
1151 P 1264 2 $XPO_GET (IOB = tsiob
1152 P 1265 2 ,PROMPT = (4, CH$PTR(UPLIT(%STRING(BELL,DEL,BELL,DEL))) )
1153 P 1266 2 ,CHARACTERS = 1
1154 1267 2 );
1155 1268 2
1156 1269 2 !Send a carriage return, so text starts at the left margin
1157 P 1270 2 $XPO_PUT (IOB = tsiob
1158 P 1271 2 ,STRING = (1, CH$PTR(UPLIT(%STRING(%CHAR(%'15')))))
1159 1272 2 );
1160 1273 1 END; !End of BWAIT
```

.PSECT \$PLITS\$,NOWRT,NOEXE,2

```
7F 07 7F 07 00048 P.AAY: .ASCII <7><127><7><127>
00 00 00 0D 0004C P.ABC: .ASCII <13><0><0><0>
```

.PSECT \$CODE\$,NOWRT,2

```
02 54 00000000G EF 9E 00002
03 53 00000000G EF 9E 00009
04 5E 08 C2 00010
02 6E 04 B0 00013
03 AE 0E 90 00016
04 AE 01 90 0C01A
04 AE 00000000' EF 9E 0001E
04 AE 00000000' EF 9E 00026
```

.ENTRY BWAIT, Save R2,R3,R4  
MOVAB XPOSFAILURE, R4  
MOVAB IOB\$+36, R3  
SUBL2 #8, SP  
MOVW #4, \$STR\$STRING  
MOVB #14, \$STR\$STRING+2  
MOVB #1, \$STR\$STRING+3  
MOVAB P.AAY, \$STR\$STRING+4  
CLRL -(SP)

1239

1267

```
04 AE 9F 00028 PUSHAB $STR$STRING
    7E D4 0002B CLRL -(SP)
00000000G EF 03 FB 0002D CALLS #3, XST$FORMAT
    52 50 D0 00034 MOVL R0, R2
    50 63 D0 00037 MOVL IOB$+36, R0
    09 13 0003A BEQL 1$
    50 DD 0003C PUSHL R0
00000000G EF 01 FB 0003E CALLS #1, XST$FREE_TEMP
    63 52 D0 00045 1$: MOVL R2, IOB$+36
    10 A3 01 B0 00048 MOVW #1, IOB$+52
    12 A3 0E 90 0004C MOVW #14, IOB$+54
    08 A3 06 90 00050 MOVW #6, IOB$+44
    DC A3 9F 00058 PUSHAB IOB$
    03 03 FB 0005B CALLS #3, XPOS$GET
    02 6E 01 B0 00062 MOVW #1, $IOB$OUTPUT
    03 AE 0E 90 00065 MOVW #14, $IOB$OUTPUT+2
    04 AE 01 90 00069 MOVW #1, $IOB$OUTPUT+3
    20 A3 00000000' EF 9E 0006D MOVAB P.ABC, $IOB$OUTPUT+4
    08 A3 6E 9E 00075 MOVAB $IOB$OUTPUT, IOB$+68
    DC A3 9F 00081 MOVW #7, IOB$+44
    00000000G EF 03 FB 00084 PUSHL R4
    7E D4 0007F CLRL -(SP)
    DC A3 9F 00081 PUSHAB IOB$
    03 FB 00084 CALLS #3, XPOS$PUT
    04 0008B RET
```

1272

1273

: Routine Size: 140 bytes, Routine Base: \$CODE\$ + 0736

```
: 1161 1274 1
: 1162 1275 1 END
: 1163 1276 0 ELUDOM
```

!End of module

## PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	68 NOVEC, WRT, RD	NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	80 NOVEC, NOWRT, RD	NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1986 NOVEC, NOWRT, RD	EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

## Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_ \$255\$DUA28:[SYSLIB]XPORT.L32;1	590	168	28	252	00:00.1



CLH  
V04-000

file processing interface and command line hand  
BWAIT -- performs user synchronization for /PAUS

1 1  
15-Sep-1984 23:56:16  
14-Sep-1984 13:05:41

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[RUNOFF.SRC]CLH.BLI;1

Page 39  
(18)

: \_\$255\$DUA28:[RUNOFF.SRC]DSRLIB.L32;1 1248 54 4 86 00:00.2

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:CLH/OBJ=OBJ\$:CLH MSRC\$:CLH/UPDATE=(ENH\$:CLH)

: Size: 1986 code + 148 data bytes  
: Run Time: 01:03.8  
: Elapsed Time: 01:54.9  
: Lines/CPU Min: 1200  
: Lexemes/CPU-Min: 73280  
: Memory Used: 465 pages  
: Compilation Complete



0337 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



0338

AH-BT13A-SE  
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY